Attachment D

```
/* iengine.c       */


#include "dmcsr.h"
#include "dmexec.h"
#include "dmfileio.h"
#include <stdio.h>
#include <malloc.h>
#include <string.h>

char FileName[MAX_FILE_NAME_SIZE];

char Zero[9] = "no alloc";
char ErrTitle[5] = "Note";
char Message[70] = "Error opening file";

MSGBOX ErrBox =
   {MSG_COMBO_OK,
          &ErrTitle[0],
                &Message[0]};

/* char FileName[30];
char Extension[4] = "DOC";
char TempName[30];
char FileBuffer1[10];

DATAFILE DataFile =
          {PERSONAL_TEXT_ASCII_FILE,
           0,
           -1,
           0,
           &FileName[0],
           &Extension[0],
           &FileBuffer1[0],
           &FileBuffer1[9],
           &FileBuffer1[9],
           &TempName[0] };

DATAFILE *pHistoryFile = &DataFile;
DATAFILE *pKBFile = &DataFile; */

MSGBOX *pErrBox = &ErrBox;


/*-------------------------------------------------------*/
/* Premise is a structure containing a single premise    */
/* of a rule. It is made up of the string itself, the    */
/* type of fact it is, and the possible values it can    */
/* hold.                                                 */
/*-------------------------------------------------------*/

struct Premise
    {
          char *pFactType;
          char *pValueSet;
          char *pValue;
         };

/*-------------------------------------------------------*/
/* Premises is a structure for maintaining a linked      */
/* list of all the premises (strings) in a single rule   */
/*-------------------------------------------------------*/

struct Premises
        {
                struct Premise *pPremise;
                int Size;
                struct Premises *pNext;
        };
```

```
  free(Variable);

  return(Result);
}
/*****************************************************************************/
int FindVariable (Variable)

/*********************************************************/
/* FindVariable searches through the fact list, looking */
/* for the variable passed in. It will then return the  */
/* value of the variable to the calling function.       */
/* This function should never fail to find a match.     */
/* If it does the Knowledge Base is flawed.             */
/*********************************************************/

char *Variable;              /* The variable to match              */

{
struct Query *pNextQuery;          /* points to the next Query          */
struct Assertion *pAssertionCopy;  /* copy of all the facts             */

int Error;                         /* error in the knowledge base       */
int NewValue;                      /* value of the variable             */

  /* initialize */

  Error = 1;
  pAssertionCopy = pAssertionList;


  /* check to see if the variable is in the fact  */
  /* list. If it is, return the value, else there */
  /* is an error in the knowledge base.           */

  while ((pAssertionList->pNext != NULL) && (Error == 1))
    {
        if (strnicmp(Variable, pAssertionList->pFact, strlen(Variable)) == 0)
          {
          sscanf(pAssertionList->pFact, " %*s %i", &NewValue);
          Error = 0;
          }
        else
          pAssertionList = pAssertionList->pNext;
        }

  pAssertionList = pAssertionCopy;

  if (Error == 1)
        {
                vid_put_string("Error in the knowledge base. No variable value for:  %s \n",
                        Variable);
                exit (0);
        }


  return(NewValue);

}
/*****************************************************************************/
```